GridIron 1.7 User's Manual

Steven J. Massey
massey@eagle.com

13 West Mercury Blvd.
Hampton, VA 23669
(757) 726-2791

November 2003

Contents

1 Introduction **1**

2 System Requirements **2**

3 Installation **2**

4 What's New **2**

5 Quick Start **2**

6 Usage **4**

7 Utility Programs **9**

8 Proposed Enhancements **11**

A Generally Patched Grid Tutorial: `simple.g` **13**

B Fortran Source for `readpatch.F` **16**

1 Introduction

GridIron is a structured grid viewer, CFD input file creator and patch database creator. Currently, CFL3D [1]/Ronnie [2] and PAB3D [3] CFD codes are supported. GridIron is compatible with any operating system capable of running X-Windows. The latest version of this manual can be obtained from the GridIron website [4].

In recent years, the number of blocks in structured grids has increased substantially. This is due to a combination of increased model geometry complexity, grid generation software behavior and subdivision of blocks for parallel processing. This has resulted in very complicated CFD input files, which are impractical to generate or modify by hand. Often, the only way the CFD input files can be created are with the original grid generation tool, which restricts changes made to the grid by other tools. This is the case when two independently generated grids are merged to form one large grid. GridIron is capable of reading a non-overset grid file and automatically creating a complete input file for CFL3D, Ronnie and PAB3D. Once, an input file is created, it may be read back into GridIron for the setting of boundary conditions and manual editing of block interface patch definitions if needed. In addition, to viewing grids and creating input files, GridIron can also create a geometrically conservative patch database for general abutting block interfaces. This type of patch database is desirable for internal flows and is currently used in PAB3D, TLNS3D [5] and Vulcan [6], all of which use the patch database creator Pao [7]. With minimal modifications, the GridIron database can be written in a compatible form for these codes. For use with an interpolation based patch database code, such as CFL3D, the solver would have to be modified.

2 System Requirements

GridIron is written in ANSI C and uses the freely available GTK+ and GLib [8] libraries for the graphical user interface. Also required is the GtkGlArea library which provides an OpenGL widget to GTK+. For Linux systems GTK+ and GLib are normally already installed and GtkGlArea will need to be installed from the distribution CD. For other Unix machines, the libraries are can be compiled from source or downloaded prebuilt in the case of SGI [9].

3 Installation

After installing the libraries discussed above, copy GridIron and the other included utility programs to a directory included in your path such as `/usr/local/bin`, then type `rehash` if your running csh or tcsh shells.

4 What's New

New since version 1.6 is the general patch cell search increment option `-i#`, where # is an integer (default=1). This option is useful to quickly generate patch input files for Ronnie or PAB3D. It can only be used with the general patching search algorithm. To generate an approximate patch input file by searching only every other 8th point - which is 64 times faster `gi -c -g -i8 grid.g`. To *quickly* generate an approximate patch input file by searching only the corner points of each face, zero may be used for the increment value, `gi -c -g -i0 grid.g`. Note, the `-i0` option is intended for generating a skeleton patch file that can then be edited by hand. If every patch is evenly divisible by #, then `-i#` will produce equivalent input files to using `-i1` (which is the default). However, the patch database will not be complete when using search increments above 1, so do not use the `-i#` option with `-p`.

5 Quick Start

For the impatient, type `gi -h` to display the usage information below, then find your application in the examples section. Note, grid files must have a `.g` extension to be recognized.

```
gi -h
GridIron 1.7.0: Built Thu Nov  6 09:49:38 EST 2003 on Linux 2.4.18-3
  Usage: gi [options] [gridfile.g] [p.cont] [cfl3d.inp ronnie.inp]
Options:
         -h       = show this help info
         -s       = single block file (default multi-block)
         -c       = run command patcher mode
         -g       = general patching (default 1:1)
         -q<#>    = general patching query level
                      1 center only
                      4 corners only (default)
                      5 center and corners
                     25 5x5 array
```

```
        -p       = generate patch database from CFL3D/Ronnie or PAB input
        -f<#.#>  = tolerance factor for 1:1 patching. Example -f0.1
        -l<#.#>  = query point length factor for general patching
        -i<#>    = general patching face cell search increment (default=1)
                   Used only with -g option and ignored with -p option.
        -v       = verbose output to stdout
Examples:
        1) To display grid: gi grid.g
        2) To display grid and CFL3D patch information: gi cfl3d.inp ronnie.inp
        3) To display grid and PAB3D patch information: gi p.cont
        4) To generate 1:1 patch input files: gi -c grid.g
        5) To generate 1:1 patch input files and find more matches:
           gi -c -f10 grid.g
        6) To generate 1:1 patch input files and find less matches:
           gi -c -f.1 grid.g
        7) To generate general patch input files: gi -c -g grid.g
        8) To generate general patch input files and handle less precise
           grid data: gi -c -g -l2 grid.g
        9) To generate general patch input files and handle more precise
           grid data: gi -c -g -l.5 grid.g
       10) To generate an approximate patch input file by searching only
           every other 8th point - which is 64x faster
           gi -c -g -i8 grid.g
       11) To QUICKLY generate an approximate patch input file by searching
           only the corner points of each face
           gi -c -g -i0 grid.g
       12) To generate a patch database for a 1:1 grid from
           previously generated Ronnie or PAB input files:
           gi -c -p clf3d.inp ronnie.inp or gi -c -p p.cont
       13) To generate a patch database for a general abutting grid from
           previously generated Ronnie or PAB input files:
           gi -c -g -p clf3dl.inp ronnie.inp or gi -c -g -p p.cont
       14) To view (do not print) a graph of every single cell patch
           ghostview patch.ps
       15) To create and view a PDF summary of every single cell patch
           psnup -256 patch.ps -pletter -d -m36 summary.ps
           ps2pdf summary.ps; acroread summary.pdf
  Notes:
        1) Grid files must have a .g extension to be recognized.
        2) General patching is not perfect so review your output.
        3) By default the code will attempt to detect machine type,
           Plot3d multi-block and precision.
        4) After patching rerun with gi p.cont and check the Show Arbitrary
           boundary conditions box for -99 to view unpatched BC's.
           Rerun with -f.5 or -f2 to decrease/increase tolerance as needed.
        5) Ronnie files are output for 1:1 and general patching.
        6) To display CFL3D input a Ronnie file must also be given.
        7) The -i0 option is intended for generating a skeleton
           patch file that can then be edited by hand.
           If every patch is evenly divisible by #, then -i# should
           be equivalent to -i1 (which is the default).
 Author: Steven J. Massey <massey@eagle.com> 2003.11.05
Version: 1.7.0, Copyright 2003 Eagle Aeronautics, Inc.
  Built: Thu Nov  6 09:49:38 EST 2003 on Linux 2.4.18-3
```

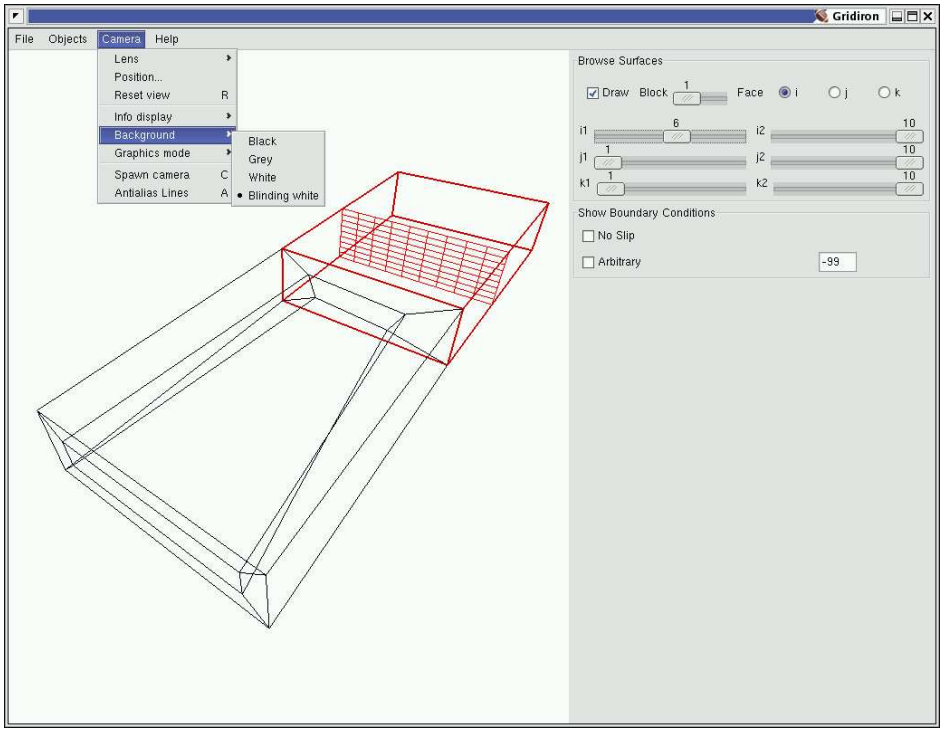


Figure 1: GUI shown in grid inspection mode with white background setting.

6 Usage

GridIron runs in three main modes: grid and/or CFD input file inspection, input file creation and patch database creation. The latter two are run in a non-graphical command line mode.

6.1 Inspection Mode

A grid or CFD input file can be passed to GridIron as an argument or read from the *File* menu item. If only a grid has been loaded, a simpler control panel for viewing i, j or k surfaces is displayed, see Figure 1. Note, that although *Show Boundary Conditions* panel is still displayed, it is not active in this mode. In the *Browse Surface* panel there are sliders for the block, face and grid coordinate ranges. When an i-face is selected the *i1* slider controls which i-plane is displayed.

When inspecting a CFL3D/Ronnie input file the GUI now has additional controls for patched and physical boundary conditions, see Figure 2. For patched faces, the source block and active area of the face is drawn in yellow and the destination block and *whole* face is drawn in purple. Note, only the destination block and face are specified, while for the source patch the block, face and cell range are required. If the face is a physical boundary condition, such as no-slip or farfield, then the face is shown in green and the *Destination* panel is not shown. In the terminology of patching, a cut represents any subset of a face. For example, a face may be made up of any number of cuts each of which can be set to different boundary conditions.

Also available when reading input files is the *Show Boundary Conditions* panel. For the *Arbitrary* type in box, use `-99` for general physical boundary conditions and PAB3D numeric codes for other boundary conditions, for example `-1` for farfield, and `-17` for general symmetry, see

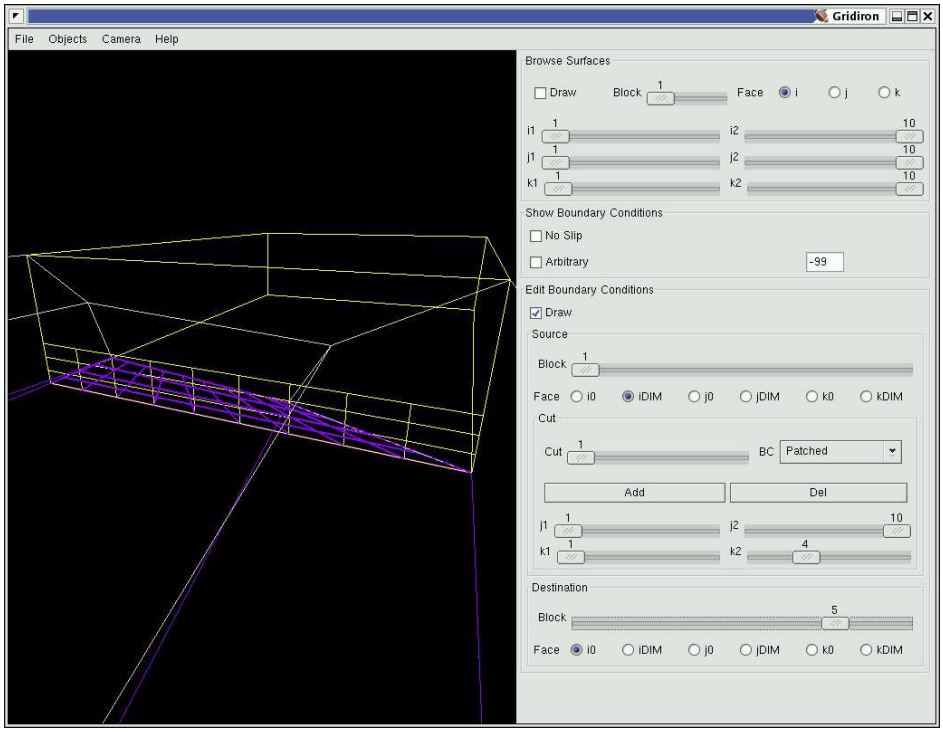


Figure 2: GUI shown in patch editing mode with the *Edit Boundary Condition* panel enabled. The source face is shown in yellow and destination is in purple.

Figure 3. Later, the *No Slip* check box will be replace with a pull down menu as in the *Edit Boundary Conditions* panel. Solid surface drawing is enabled by the `s` hot-key and multiple viewing windows can be spawned with the `c` hot-key, see Figure 4.

6.2 Patch File Creation Mode

CFD solver and patch input files can be created for any type of multiblock abutting grids. A special routine is available for 1:1 point matching interfaces, but Ronnie input files are always generated, since they work for 1:1 or general patches. Access to the patch file creation process is only from the command line interface.

6.2.1 1:1 Point Matched Grids

Point matched grid patching is determined by comparing matching up cell face centers to within a tolerance proportional to the square root of the cell face area. This tolerance may be adjusted by a user specified factor using the `-f#.#` option, where a number greater than one increases the allowable distance between two points that would be considered to be at the same location and a factor less than one would decrease this distance, thus reducing the number of possible matches. To generate input files for a point matched grid named `vg.g` type `gi -c vg.g` which gives:

```
GridIron 1.7.0: Built Thu Nov  6 09:49:38 EST 2003 on Linux 2.4.18-3
Total grid points = 14835
   1:    10,    18,    19
```

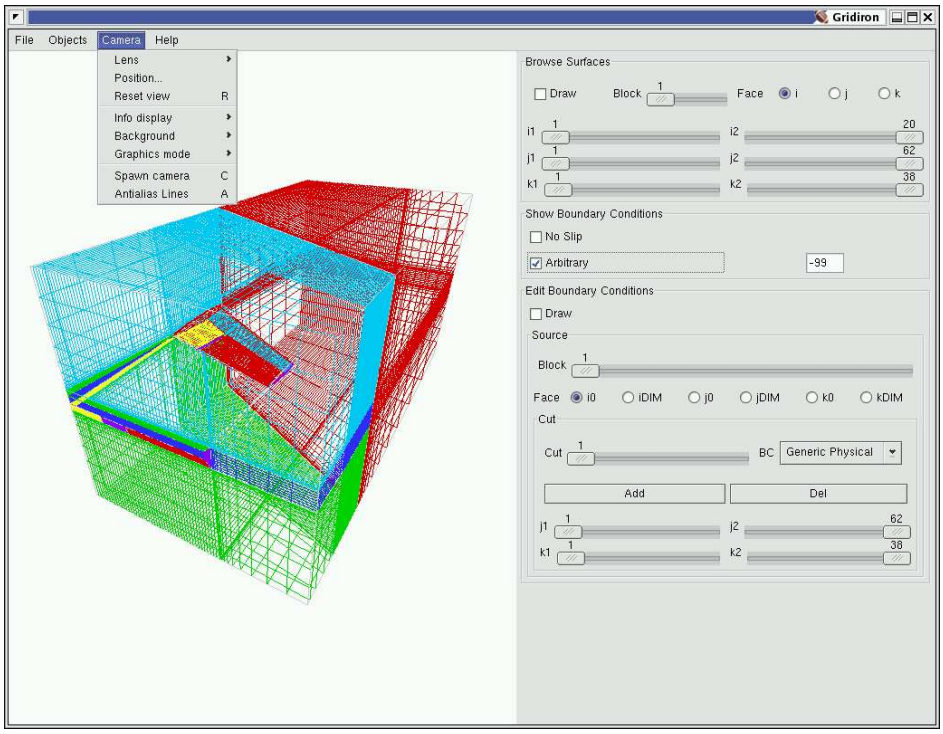


Figure 3: *Show Boundary Conditions* panel enabled displaying the default code for unspecified physical boundary conditions.

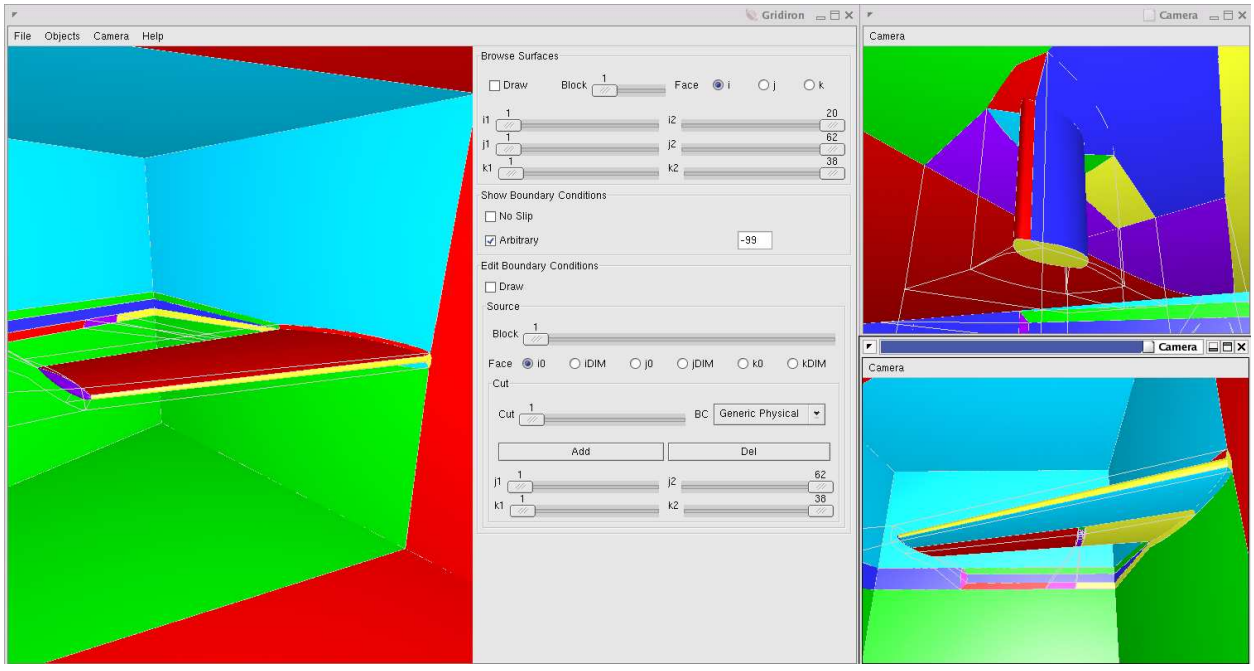


Figure 4: *Show Boundary Conditions* panel enabled with solid surfaces and multiple camera views.

```
   2:    19,     5,    19
   3:    19,    14,    10
   4:    10,    10,    10
   5:    19,     5,    10
   6:    10,    10,    10
   7:    10,    10,    10
   8:    10,    10,    10
   9:    10,    10,    10
  10:    10,    10,    10
File vg.g read in  0.03 seconds ( 0.46 Mp/s) (10.55 MB/s)
  1: X[ 4.53e+02,  5.02e+02]  Y[ 3.47e+02,  3.73e+02]  Z[-6.44e+01, -3.39e+01]
  2: X[ 4.90e+02,  5.28e+02]  Y[ 3.47e+02,  3.56e+02]  Z[-6.42e+01, -3.34e+01]
  3: X[ 4.94e+02,  5.35e+02]  Y[ 3.55e+02,  3.73e+02]  Z[-6.41e+01, -4.50e+01]
  4: X[ 5.08e+02,  5.30e+02]  Y[ 3.56e+02,  3.62e+02]  Z[-4.55e+01, -3.33e+01]
  5: X[ 4.97e+02,  5.35e+02]  Y[ 3.62e+02,  3.73e+02]  Z[-4.55e+01, -3.32e+01]
  6: X[ 4.94e+02,  4.99e+02]  Y[ 3.55e+02,  3.62e+02]  Z[-4.56e+01, -3.40e+01]
  7: X[ 5.06e+02,  5.10e+02]  Y[ 3.56e+02,  3.62e+02]  Z[-4.55e+01, -3.38e+01]
  8: X[ 4.94e+02,  5.08e+02]  Y[ 3.55e+02,  3.58e+02]  Z[-4.56e+01, -3.39e+01]
  9: X[ 4.97e+02,  5.10e+02]  Y[ 3.60e+02,  3.63e+02]  Z[-4.55e+01, -3.38e+01]
 10: X[ 4.94e+02,  5.10e+02]  Y[ 3.55e+02,  3.63e+02]  Z[-4.56e+01, -4.39e+01]
Determining 1:1 patching for all faces.
Number of cell searches per block/face:
 BLOCK        I0      IDIM        J0      JDIM        K0      KDIM   TIME(s)
     1    669222     69618    306990    317844    317547    310833      0.05
     2      1092    185112    710532     51188    187164    172098      0.03
     3      3703    309231      2697    357534    571914    119472      0.06
     4      2942    145881      5702      3316      4245    219267      0.01
     5      3637     68850     36243    209898      7600    160110      0.02
     6      4156      3922      4966    334575      3943    297351      0.02
     7      4405      3787      5064    324405    370089      1998      0.02
     8      5128    352107      2969      2230      4803    321975      0.02
     9      5431    322083      2671      3140    306909      2772      0.02
    10      4217      4103      4481      4102      4250    447867      0.02
0 extra patch warnings.
0 missing patch warnings.
Total Run Time  0.83 seconds
```

and produces CFL3D input files `cfl3d.inp, ronnie.inp` as well as PAB3D files `p.cont` and `user.cont`. If warnings are issued, rerun with the `-f#.#` option with $f < 1$ if there are any extra patch warnings. Then run in edit mode to address any remaining warnings or to fill in the correct physical boundary conditions. For example, type `gi cfl3d.inp ronnie.inp` or `gi p.cont` (`user.cont` is automatically read as specified in `p.cont`).

6.2.2 General Patch Grids

General patch file creation is based on point-in-polyhedron [10] checking of off surface points normal to a source cell face, see Figure 5. The distance away from the source cell face is proportional to the cell volume divided by its area and may be scaled using the `-l#.#` option. If GridIron produces extra patch warnings, rerun with a smaller length factor (less than one) until GridIron no further errors can be eliminated. If missing patch errors are introduced, increase the length factor. If a point just outside of an cell face is inside another cell volume, then that face is communicating

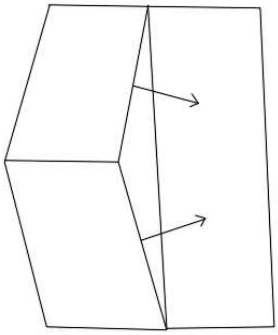
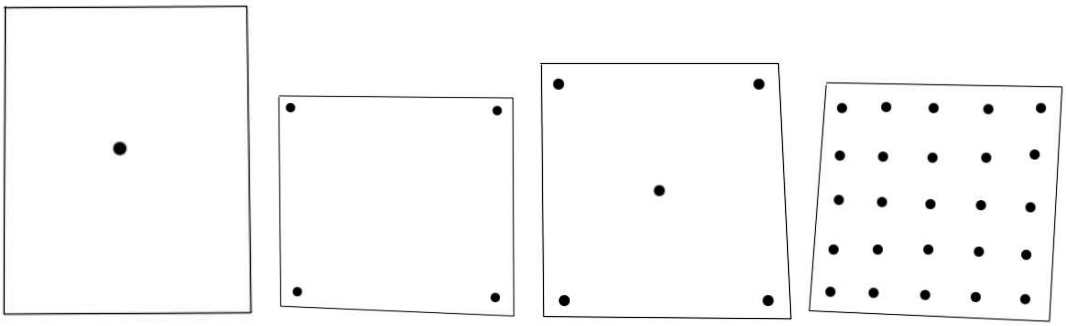


Figure 5: Query points normal and away from source cells for general patching of gapped cells.



(a) `-q1` (b) `-q4` (c) `-q5` (d) `-q25`

Figure 6: Optional query point patterns for general patching, `-q4` is default.

with the face of the destination block face and cell for which the cell is a part of. In the case of the redundant destination cells that are on block edges and corners, the destination face is determined by choosing that cell face whose normal is most parallel to the source face normal. In simple terms, if a cell can reach out and touch another cell, then they are talking to each other. In this phase, the query points are only designed to determine who each cell is talking to, but not to what degree. In the patch database phase the exact area overlap between source and destination cells is determined. The command line option `-q#` allows for adjusting the number of query points generated for each cell from 1 to 25 insure all communicating cells are located, see Figure 6.

6.3 Patch Database Creation Mode

In the patch database creation mode, GridIron is supplied with input file that has been checked for errors and omissions. GridIron then searches the grid within the specified ranges to resolve each cell's communication with other cells. This communication is characterized by an area balance. Currently the database is written in c-binary (`patchdb.bin`), Fortran unformatted (`patchdb.unf`) and formatted ASCII text (`patchdb.fmt`). GridIron uses the compiler default byte ordering for binary output, which for gcc on Linux is little endian. As a debugging tool, the total patch database is output in graphical form in a PostScript file `patch.ps`, where each

page represents a single cell to cell communication. Do not print this file as it can be extremely long. To conveniently view this file run the following commands:

```
/usr/TeX/bin/i386-linux/psnup -256 patch.ps -pletter -d -m36 summary.ps
/usr/bin/pspdf summary.ps
/usr/local/Acrobat5/bin/acroread summary.pdf
```

You may need to install these free helper programs on your system to produce the summary pdf file. In Figure 7, a sample cell graph is shown for a single source cell with eight destination cells. The graph is helpful in determining the cause of any detected area discrepancies. The output is ordered by source cell b, f, r, s address where b = block, f = face number (1=imin, 2=imax, 3=jmin, 4=jmax, 5=kmin, 6=kmax) and local r, s coordinate which is i, j or k depending on the face.

6.3.1 Patch Database Structure

The patch database contains cell connectivity and boundary condition information in an array called `cell_info`. For patched cells, an source and destination area array called `area_info` is provided. An example Fortran code (`readpatch.F`) is provided that reads the entire database (`patchdb.unf`) and writes it in unformatted binary(`out.unf`), formatted ASCII(`out.fmt`), and human readable ASCII(`out.info`). The read statements are as follows and the whole code is in Appendix B.

7 Utility Programs

Included in the GridIron package are several utilities to translate and combine the CFD input files and grids.

7.1 `cfl2pab`

GridIron 's native read format is PAB3D, so whenever a CFL3D/Ronnie input files are read, GridIron calls `cfl2pab` to convert them for reading. `cfl2pab` is a stand alone program that maybe run from the command line. On-line help is available by the `-h` option as follows:

```
% cfl2pab -h

Purpose: Convert CFL3D input files to PAB3D.
  Usage: cfl2pab [options] cfl3d.inp ronnie.inp [pab3d.cont] [user.cont]
Options:
         -h       = show this help info
  Notes: 1) This program expects patching data to be in ronnie.inp.
         2) ronnie.inp is expected to be in format 0.
Example: cfl2pab cfl3d.inp ronnie.inp p.cont u.cont
         reads CFL3D input files cfl3d.inp ronnie.inp and writes
         PAB3D input files p.cont u.cont.
 Author: Steven J. Massey <massey@eagle.com> 2003.07.02
Version: 1.0.1, Copyright 2003 Eagle Aeronautics, Inc.
  Built: Wed Jul  2 12:30:23 EDT 2003
     OS: Linux 2.4.18-14
```

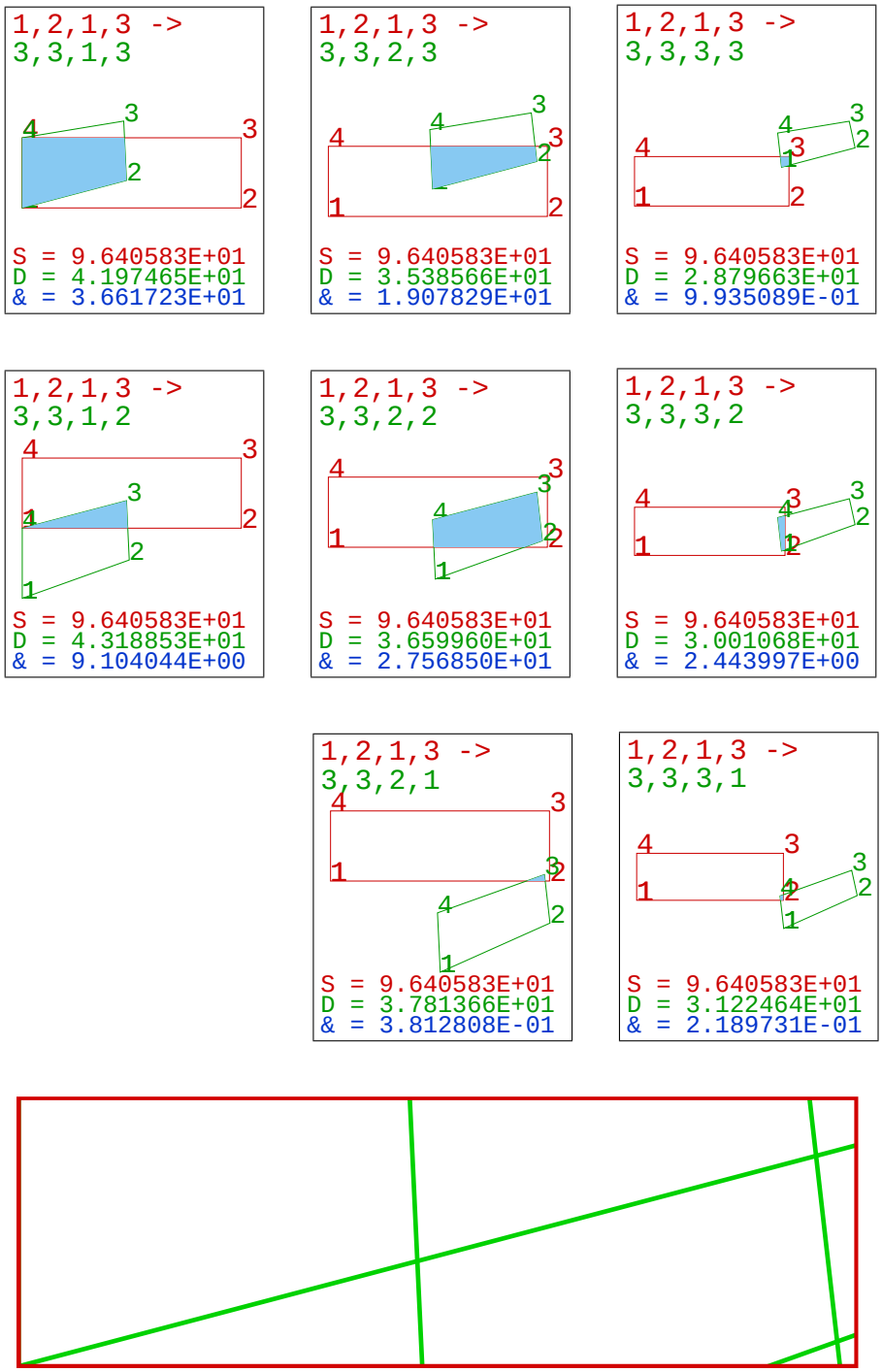


Figure 7: Sample cell graph pages for a single source cell (*red*) from block 1, face 2, $j = 1$, $k = 3$ communicating with block 3, face 3 (*green*). Cells are drawn to the local page scale and the overlapping areas are shown in blue. Numbers represent the projected areas of the source, destination and overlap respectively. Also shown is the source cell assembled from the eight destinations.

7.2 pab2cfl

When interactive changes to the boundary/patch conditions are saved in CFL3D format, GridIron writes temporary PAB3D files and calls `pab2cfl` to convert them. In patch file generation mode, GridIron creates the PAB3D and CFL3D/Ronnie files independently based on the internal patch database. `pab2cfl` is a stand alone program that maybe run from the command line. On-line help is available by the `-h` option as follows:

```
%pab2cfl -h

PAB2CFL V1.1.0 20030730 massey@eagle.com

usage: pab2cfl [options] tpab3.cont [cfl3d.inp] [ronnie.inp]
options are :
   -h       = show this help info
```

7.3 catgrid

To merge a set of CFL3D input files and grids together a utility called `catgrid` is provided. On-line usage information is displayed the `-h` option as follows:

```
%catgrid -h

Purpose: Concatenate multiblock, plot3d, unformatted Fortran grids and
         optionally CFL3D input files.
  Usage: catgrid [options] file_in1 file_in2 [file_in3 ...] file_out
Options:
         -h       = show this help info
         -v       = verbose output
         -c       = concatenate CFL3D input files and their grids
         -M<mach> = Machine type for binary or unformatted - one of:
                    ieee  bsieee  iris  alpha   hp  ibm
                    sun   dec     cray  convex  nt  linux
                    Case is not significant and only the first
                    2 characters are needed. (default auto-detected)
  Notes: 1) By default the code will detect machine type,
            Plot3d multi-block and single or double precision.
         2) If -M is used, all grids must be in the same binary format,
            but do *not* have to be all single or multiblock.
         3) Output will only be in IEEE (big or little endian)
            unformatted Fortran.
 Author: Steven J. Massey <massey@eagle.com> 2002.12.18
Version: 1.0.0, Copyright 2002 Eagle Aeronautics, Inc.
  Built: Sat Feb  8 19:59:19 EST 2003
     OS: Linux 2.2.17-14
```

8 Proposed Enhancements

During the development process many ideas for optimization were realized, but not implemented because of time constraints and the desire not to over complicate the code before the core routines

were stabilized. It is hoped that as time permits the following enhancements will be made to further increase the usefulness of GridIron.

8.1 Automatic Locally Adaptive Searching

Currently all code parameters are set globally from either user specified arguments or internal code parameters. Because of the wide variation of cell spacing and topology within a single grid, it is desirable to have the code automatically adjust its search parameters based on the detection of errors in cell area conservation on a per cell basis. In addition to simple tolerances and distances, the number of query points and the search algorithm itself could be changed on the fly from a coarse to fine search until the area conservation criteria is met. This would greatly decrease the time required for the code to run and at the same time simplify its usage.

8.2 Code Parallelization

GridIron runs on a single processor with run times (on a 2 GHz Pentium 4) from a few minutes for most two-dimensional grids to as much as 15 hours for a generally patched 140 block - 10 million cell grid. Because the search algorithm is completely independent down to the cell level, the work may be spread between an arbitrary number of processors with high efficiency. Since CFD on large grids is routinely run on Beowulf clusters, it would be convenient to also run the patching on the same cluster of computers. This is also useful because, GridIron has tolerance parameters that may need to be adjusted to maximize the accuracy of the generated input file. Thus, being able to rerun GridIron quickly, rather than overnight, provides the engineer with quick verification of the patching process. Modifying GridIron to use the industry standard MPICH parallel library is a straight forward way to insure that GridIron runs as fast as possible.

8.3 Improved Patch File Output

In the current implementation, all patching information is written in the original Ronnie input format. Because of the inherent complexity of the patching process, it is desirable to provide Ronnie with the maximum amount of information thus, it is proposed that GridIron be modified to output the new ”version 1” of the Ronnie input file which contains additional index information on all destination patches. Furthermore, since it is not necessary to run Ronnie for one to one topologies, it is proposed that GridIron be modified to write the complete one to one patching information inside the CFL3D input file.

8.4 CFD Related Graphical User Interface Functions

The primary mission of GridIron is to automatically generate input files and the patch databases that contains all the grid connectivity information and allow the user to specify rudimentary boundary condition information. In addition, it would be helpful to be able to control all settings available in the CFD input file from the GUI. The author participated in a Phase II SBIR [11] project which included a code written in Java that provided a full featured GUI for CFL3D and PAB3D. A similar design could be implemented in GridIron in ANSI C.

References

[1] Krist, S., Biedron, R., and Rumsey, C., "CFL3D User's Manual (Version 5.0)," NASA/TM 1998-208444, 1998.

[2] Biedron, R. and Thomas, J., "A Generalized Patched-Grid Algorithm with Application to the F-18 Forebody with Actuated Control Strake," *Computing Systems in Engineering*, Vol. 1, No. 2, 1990, pp. 563–576.

[3] Abdol-Hamid, K., "PAB3D Manual," `http://www.asm-usa.com/software/PAB3D/PAB3D-Doc.html`.

[4] The GridIron website, `http://eagle.com/gridiron`.

[5] Vatsa, V. and Wedan, B., "Development of a Multigrid Code for 3-D Navier-Stokes Equations and Its Application to a Grid-Refinement Study," *Computers and Fluids*, Vol. 18, 1990, pp. 391–403.

[6] White, J., "VULCAN User's Manual," `http://vulcan-cfd.larc.nasa.gov/WebPage_manual/vulcan_manual.html`.

[7] Abdol-Hamid, K. S., Carlson, J. R., and Pao, S. P., "Calculation of Turbulent Flows Using Mesh Sequencing and Conservative Patch Algorithm," AIAA Paper 95-2336, 1995.

[8] The GIMP Toolkit website, `http://gtk.org`.

[9] SGI Freeware Libraries, `http://freeware.sgi.com`.

[10] O'Rourke, J., *Computational Geometry in C*, Cambridge University Press, 1998.

[11] Abdol-Hamid, K. S., Massey, S., and Caldwell, S., "Unified Process Management System for Computational Fluid Dynamics (UPMS)," AIAA Paper 2003-0803, 2003.

A Generally Patched Grid Tutorial: `simple.g`

1. To inspect the grid:

   ```
   gi simple.g
   ```

2. To create CFD input files from grid:

   ```
   gi -c -g -l0.1 simple.g
   ```

3. To edit the CFD input files to check ranges and cuts or to set boundary condition types interactively:

   ```
   gi cfl3d.inp ronnie.inp
   ```

   or if you want to work with the PAB3D files you can type

   ```
   gi p.cont
   ```

   Once you have loaded the input files you can save them in either format.

4. If there were no user changes made to the patcher input file, then the patchdb files are in the final form. If changes were made then rerun `gi` with the patch input files:

   ```
   gi -c -g -l0.1-p cfl3d.inp ronnie.inp
   ```

   Be consistent in the usage of options for Steps 3 and 4. ***When you use the `-l#.#` option in Step 3, use the same option in Step 4***. The files `cfl3d.check.inp` and `ronnie.check.inp` are what GridIron used internally to create the patch database. If they differ from the input files, the input files may have user specified communications that do not exist or communications that were not detected.

A.1 Terminal Session for Steps 1-4

```
% ls -l
total 64
-rw-rw-r--    1 massey   massey      60140 Oct 29 13:31 simple.g
% gi simple.g
GridIron 1.7.0: Built Thu Nov  6 09:49:38 EST 2003 on Linux 2.4.18-3
Total grid points = 2500
   1:    10,    10,    10
   2:     5,    10,    10
   3:    10,     5,     5
   4:    10,     5,     5
   5:     5,    10,     5
   6:     5,    10,     5
File simple.g read in  0.02 seconds ( 0.11 Mp/s) ( 2.41 MB/s)
  1: X[ 7.62e+01,  3.13e+02]  Y[ 5.34e+01,  2.15e+02]  Z[ 0.00e+00,  5.00e+01]
  2: X[ 3.09e+02,  5.45e+02]  Y[ 7.57e+01,  2.02e+02]  Z[ 1.04e+01,  3.50e+01]
  3: X[ 3.08e+02,  5.41e+02]  Y[ 5.70e+01,  9.18e+01]  Z[ 0.00e+00,  5.00e+01]
  4: X[ 3.11e+02,  5.45e+02]  Y[ 1.61e+02,  2.17e+02]  Z[ 0.00e+00,  5.00e+01]
  5: X[ 3.08e+02,  5.45e+02]  Y[ 5.70e+01,  2.17e+02]  Z[ 0.00e+00,  1.56e+01]
  6: X[ 3.08e+02,  5.45e+02]  Y[ 5.70e+01,  2.17e+02]  Z[ 3.00e+01,  5.00e+01]
% gi -c -g -l0.1 simple.g
GridIron 1.7.0: Built Thu Nov  6 09:49:38 EST 2003 on Linux 2.4.18-3
Total grid points = 2500
   1:    10,    10,    10
   2:     5,    10,    10
   3:    10,     5,     5
   4:    10,     5,     5
   5:     5,    10,     5
   6:     5,    10,     5
File simple.g read in  0.00 seconds ( 1.99 Mp/s) (45.56 MB/s)
  1: X[ 7.62e+01,  3.13e+02]  Y[ 5.34e+01,  2.15e+02]  Z[ 0.00e+00,  5.00e+01]
  2: X[ 3.09e+02,  5.45e+02]  Y[ 7.57e+01,  2.02e+02]  Z[ 1.04e+01,  3.50e+01]
  3: X[ 3.08e+02,  5.41e+02]  Y[ 5.70e+01,  9.18e+01]  Z[ 0.00e+00,  5.00e+01]
  4: X[ 3.11e+02,  5.45e+02]  Y[ 1.61e+02,  2.17e+02]  Z[ 0.00e+00,  5.00e+01]
  5: X[ 3.08e+02,  5.45e+02]  Y[ 5.70e+01,  2.17e+02]  Z[ 0.00e+00,  1.56e+01]
  6: X[ 3.08e+02,  5.45e+02]  Y[ 5.70e+01,  2.17e+02]  Z[ 3.00e+01,  5.00e+01]
Determining general patching for all faces.
Number of cell searches per block/face:
 BLOCK        I0      IDIM        J0      JDIM        K0      KDIM   TIME(s)
     1         0      9456         0         0         0         0      0.06
     2      4105         0      1380      1484       884       866      0.05
```

```
     3       299       319       919         0         0      1007      0.03
     4       388       303      2253         0      1504         0      0.05
     5      1722         0       404       353         0       836      0.03
     6      2455         0       444       411       935         0      0.03
0 extra patch warnings.
0 missing patch warnings.
Total Run Time  0.54 seconds
% ls -l
total 3168
-rw-rw-r--    1 massey   massey       9286 Oct 29 13:33 cfl3d.inp
-rw-rw-r--    1 massey   massey      48004 Oct 29 13:33 patchdb.bin
-rw-rw-r--    1 massey   massey      77108 Oct 29 13:33 patchdb.fmt
-rw-rw-r--    1 massey   massey      48052 Oct 29 13:33 patchdb.unf
-rw-rw-r--    1 massey   massey    2958399 Oct 29 13:33 patch.ps
-rw-rw-r--    1 massey   massey       6600 Oct 29 13:33 p.cont
-rw-rw-r--    1 massey   massey       4181 Oct 29 13:33 ronnie.inp
-rw-rw-r--    1 massey   massey      60140 Oct 29 13:31 simple.g
-rw-rw-r--    1 massey   massey       2404 Oct 29 13:33 user.cont
% gi -c -g -p -l0.1 cfl3d.inp ronnie.inp
GridIron 1.7.0: Built Thu Nov  6 09:49:38 EST 2003 on Linux 2.4.18-3
expecting cfl3d and ronnie input files
cfl3d file = cfl3d.inp, ronnie file = ronnie.inp
cfl2pab  cfl3d.inp ronnie.inp .p.cont .u.cont
Associated Grid File = simple.g
Associated user.cont File = .u.cont
Total grid points = 2500
   1:    10,    10,    10
   2:     5,    10,    10
   3:    10,     5,     5
   4:    10,     5,     5
   5:     5,    10,     5
   6:     5,    10,     5
File simple.g read in  0.00 seconds ( 1.81 Mp/s) (41.40 MB/s)
  1: X[ 7.62e+01,  3.13e+02]  Y[ 5.34e+01,  2.15e+02]  Z[ 0.00e+00,  5.00e+01]
  2: X[ 3.09e+02,  5.45e+02]  Y[ 7.57e+01,  2.02e+02]  Z[ 1.04e+01,  3.50e+01]
  3: X[ 3.08e+02,  5.41e+02]  Y[ 5.70e+01,  9.18e+01]  Z[ 0.00e+00,  5.00e+01]
  4: X[ 3.11e+02,  5.45e+02]  Y[ 1.61e+02,  2.17e+02]  Z[ 0.00e+00,  5.00e+01]
  5: X[ 3.08e+02,  5.45e+02]  Y[ 5.70e+01,  2.17e+02]  Z[ 0.00e+00,  1.56e+01]
  6: X[ 3.08e+02,  5.45e+02]  Y[ 5.70e+01,  2.17e+02]  Z[ 3.00e+01,  5.00e+01]
Determining general patching for all faces.
Number of cell searches per block/face:
 BLOCK        I0      IDIM        J0      JDIM        K0      KDIM   TIME(s)
     1         0     19327         0         0         0         0      0.06
     2      3946         0      1367      1448       845       830      0.05
     3       299       319       911         0         0       918      0.05
     4       349       303      1980         0      1588         0      0.02
     5      1545         0       388       353         0       830      0.02
     6      2248         0       444       488       833         0      0.02
0 extra patch warnings.
0 missing patch warnings.
Total Run Time  0.54 seconds
% ls -l
total 3196
-rw-rw-r--    1 massey   massey       9286 Oct 29 13:36 cfl3d.check.inp
```

```
-rw-rw-r--    1 massey   massey       9286 Oct 29 13:33 cfl3d.inp
-rw-rw-r--    1 massey   massey      48004 Oct 29 13:36 patchdb.bin
-rw-rw-r--    1 massey   massey      77108 Oct 29 13:36 patchdb.fmt
-rw-rw-r--    1 massey   massey      48052 Oct 29 13:36 patchdb.unf
-rw-rw-r--    1 massey   massey    2958399 Oct 29 13:36 patch.ps
-rw-rw-r--    1 massey   massey       6600 Oct 29 13:36 p.check.cont
-rw-rw-r--    1 massey   massey       6600 Oct 29 13:33 p.cont
-rw-rw-r--    1 massey   massey       4181 Oct 29 13:36 ronnie.check.inp
-rw-rw-r--    1 massey   massey       4181 Oct 29 13:33 ronnie.inp
-rw-rw-r--    1 massey   massey      60140 Oct 29 13:31 simple.g
-rw-rw-r--    1 massey   massey       2404 Oct 29 13:33 user.cont
% psnup -256 patch.ps -pletter -d -m36 summary.ps
[1] [2] [3] [4] [5] [6] [7] Wrote 7 pages, 3686751 bytes
% ps2pdf summary.ps
% acroread summary.pdf
% ls -l
total 7144
-rw-rw-r--    1 massey   massey       9286 Oct 29 13:36 cfl3d.check.inp
-rw-rw-r--    1 massey   massey       9286 Oct 29 13:33 cfl3d.inp
-rw-rw-r--    1 massey   massey      48004 Oct 29 13:36 patchdb.bin
-rw-rw-r--    1 massey   massey      77108 Oct 29 13:36 patchdb.fmt
-rw-rw-r--    1 massey   massey      48052 Oct 29 13:36 patchdb.unf
-rw-rw-r--    1 massey   massey    2958399 Oct 29 13:36 patch.ps
-rw-rw-r--    1 massey   massey       6600 Oct 29 13:36 p.check.cont
-rw-rw-r--    1 massey   massey       6600 Oct 29 13:33 p.cont
-rw-rw-r--    1 massey   massey       4181 Oct 29 13:36 ronnie.check.inp
-rw-rw-r--    1 massey   massey       4181 Oct 29 13:33 ronnie.inp
-rw-rw-r--    1 massey   massey      60140 Oct 29 13:31 simple.g
-rw-rw-r--    1 massey   massey     341518 Oct 29 13:37 summary.pdf
-rw-rw-r--    1 massey   massey    3686751 Oct 29 13:37 summary.ps
-rw-rw-r--    1 massey   massey       2404 Oct 29 13:33 user.cont
```

B Fortran Source for `readpatch.F`

```
      program readpatch
c
c S. J. Massey, Eagle Aeronautics, Inc.
c massey@eagle.com
c
c Example Fortran code to read gridiron patchdb.unf (v 031006).
c For C-code, see gridiron/batch_main.c/write_patchdb().
c
c Default byte order for gcc-linux and hence gridiron is little endian.
c For Lahey clear FORT90L variable as follows: setenv FORT90L.
c For ifc, setenv F_UFMTENDIAN little.
c
      implicit none
      integer nbl, nfaces
      integer n,f,r,s,c
      integer rmax,smax,cmax

      parameter (nfaces=6)
```

```
      integer, allocatable :: rsizes(:,:), ssizes(:,:)
      integer, allocatable :: cell_info_sizes(:,:,:,:)
      integer, allocatable :: cell_info(:,:,:,:,:)
      real*8, allocatable :: area_info(:,:,:,:,:)

      open(1,file='patchdb.unf',form='unformatted')
      open(2,file='out.unf',form='unformatted')
      open(3,file='out.fmt',form='formatted')
      open(4,file='out.info',form='formatted')
      read(1) nbl

      allocate (rsizes(nfaces,nbl))
      allocate (ssizes(nfaces,nbl))
      read(1) ((rsizes(f,n),f=1,nfaces),n=1,nbl)
      read(1) ((ssizes(f,n),f=1,nfaces),n=1,nbl)

c Find max array size (for padded array allocation).
c Could alternatively use single dimensional array and compute index
c or allocate an array of pointers to handle nonuniform array dimensions.
      rmax = 0
      smax = 0
      do n = 1, nbl
        do f = 1, nfaces
          if (rsizes(f,n).gt.rmax) rmax = rsizes(f,n)
          if (ssizes(f,n).gt.smax) smax = ssizes(f,n)
        enddo
      enddo
      allocate (cell_info_sizes(smax,rmax,nfaces,nbl))
      read(1) ((((cell_info_sizes(s,r,f,n),s=1,ssizes(f,n)),
     .            r=1,rsizes(f,n)),f=1,nfaces),n=1,nbl)

c Find max cell_info_size for padded array allocation

      cmax = 0
      do n = 1, nbl
        do f = 1, nfaces
          do r = 1, rsizes(f,n)
            do s = 1, ssizes(f,n)
              if (cell_info_sizes(s,r,f,n).gt.cmax)
     .          cmax = cell_info_sizes(s,r,f,n)
            enddo
          enddo
        enddo
      enddo

      allocate (cell_info(cmax,smax,rmax,nfaces,nbl))
      read(1) (((((cell_info(c,s,r,f,n),
     .             c=1,cell_info_sizes(s,r,f,n)),
     .             s=1,ssizes(f,n)),
     .             r=1,rsizes(f,n)),
     .             f=1,nfaces),
     .             n=1,nbl)

      allocate (area_info(cmax+1,smax,rmax,nfaces,nbl))
```

```
      read(1) (((((area_info(c,s,r,f,n),
     .             c=1,cell_info_sizes(s,r,f,n)+1),
     .             s=1,ssizes(f,n)),
     .             r=1,rsizes(f,n)),
     .             f=1,nfaces),
     .             n=1,nbl)

      write(2) nbl
      write(2) ((rsizes(f,n),f=1,nfaces),n=1,nbl)
      write(2) ((ssizes(f,n),f=1,nfaces),n=1,nbl)
      write(2) ((((cell_info_sizes(s,r,f,n),s=1,ssizes(f,n)),
     .             r=1,rsizes(f,n)),f=1,nfaces),n=1,nbl)
      write(2) (((((cell_info(c,s,r,f,n),
     .             c=1,cell_info_sizes(s,r,f,n)),
     .             s=1,ssizes(f,n)),
     .             r=1,rsizes(f,n)),
     .             f=1,nfaces),
     .             n=1,nbl)
      write(2) (((((area_info(c,s,r,f,n),
     .             c=1,cell_info_sizes(s,r,f,n)+1),
     .             s=1,ssizes(f,n)),
     .             r=1,rsizes(f,n)),
     .             f=1,nfaces),
     .             n=1,nbl)

c
c output fortran formatted ascii
c
      write(3,*) nbl
      write(3,*) ((rsizes(f,n),f=1,nfaces),n=1,nbl)
      write(3,*) ((ssizes(f,n),f=1,nfaces),n=1,nbl)
      write(3,*) ((((cell_info_sizes(s,r,f,n),s=1,ssizes(f,n)),
     .             r=1,rsizes(f,n)),f=1,nfaces),n=1,nbl)
      write(3,*) (((((cell_info(c,s,r,f,n),
     .             c=1,cell_info_sizes(s,r,f,n)),
     .             s=1,ssizes(f,n)),
     .             r=1,rsizes(f,n)),
     .             f=1,nfaces),
     .             n=1,nbl)
      write(3,*) (((((area_info(c,s,r,f,n),
     .             c=1,cell_info_sizes(s,r,f,n)+1),
     .             s=1,ssizes(f,n)),
     .             r=1,rsizes(f,n)),
     .             f=1,nfaces),
     .             n=1,nbl)
c
c output human readable formatted ascii
c
      write(4,'(a)') "# Human readable gridiron patch database"
      write(4,'("nbl = ",i10)') nbl
      write(4,'(a)') "# rsizes(face,block)"
      do n = 1, nbl
        do f = 1, nfaces
          write(4,'("rsizes(",i5,",",i5,") = ",i10)') f,n,rsizes(f,n)
```

```
  enddo
 enddo
 write(4,'(a)') "# ssizes(face,block)"
 do n = 1, nbl
   do f = 1, nfaces
     write(4,'("ssizes(",i5,",",i5,") = ",i10)') f,n,ssizes(f,n)
   enddo
 enddo
 write(4,'(a)') "# cell_info_sizes(s,r,face,block)"
 do n = 1, nbl
   do f = 1, nfaces
     do r = 1, rsizes(f,n)
       do s = 1, ssizes(f,n)
         write(4,
.       '("cell_info_sizes(",i5,",",i5,",",i5,",",i5,") = ",i10)')
.        s,r,f,n,cell_info_sizes(s,r,f,n)
       enddo
     enddo
   enddo
 enddo

 write(4,'(a)') "# cell_info(c,s,r,face,block)"
 do n = 1, nbl
   do f = 1, nfaces
     do r = 1, rsizes(f,n)
       do s = 1, ssizes(f,n)
         do c = 1, cell_info_sizes(s,r,f,n)
           write(4,
.     '("cell_info(",i5,",",i5,",",i5,",",i5,",",i5,") = ",i10)')
.        c,s,r,f,n,cell_info(c,s,r,f,n)
         enddo
       enddo
     enddo
   enddo
 enddo

 write(4,'(a)') "# area_info(c,s,r,face,block)"
 do n = 1, nbl
   do f = 1, nfaces
     do r = 1, rsizes(f,n)
       do s = 1, ssizes(f,n)
         do c = 1, cell_info_sizes(s,r,f,n)+1
           write(4,
.     '("area_info(",i5,",",i5,",",i5,",",i5,",",i5,") = ",e15.7)')
.        c,s,r,f,n,area_info(c,s,r,f,n)
         enddo
       enddo
     enddo
   enddo
 enddo
 end
```